10.48047/jocaaa.2024.33.08.269

Real-Time Multi-Mode Hand Gesture Recognition Using MediaPipe and Deep Learning for Human-Computer Interaction

Anil Khatak[1], Sheenam Naaz[2],

[1] Department of Allied Health Sciences, Guru Jambheshwar University of Science and Technology, Hisar, India, 125001.

[2]Department of Computer Science and Engineering, SSCSE, Sharda University, Greater Noida, India, 201310.

Abstract

This research presents a real-time multi-mode hand gesture recognition system designed to enable touchless human-computer interaction using only a standard webcam. The proposed system integrates Google's MediaPipe Hands framework for efficient hand landmark detection and a lightweight neural network classifier developed with TensorFlow for gesture recognition. It supports three operational modes: cursor control, presentation navigation, and teaching aid interaction, each mapped to intuitive hand gestures such as pinching, swiping, or pointing. The modular architecture facilitates a smooth transition between modes, with gesture-to-command mapping implemented through a user-friendly Tkinter interface. To ensure responsiveness and reliability, the system employs data preprocessing, exponential smoothing for jitter-free tracking, and real-time feedback overlays. Experimental results demonstrate sub-50 millisecond latency and over 90% classification accuracy, even under variable lighting and background conditions. Case studies in smart classrooms and creative workspaces highlight its practical utility, confirming its potential for widespread deployment in education, remote collaboration, and accessibility focused applications. The system operates on low-resource consumer-grade hardware, promoting cost-effective scalability and ease of use. Future enhancements include support for dynamic gestures, user-defined customization, and IoT or AR integrations.

Keywords: AI, Gesture Recognition ·Media Pipe ·Human Computer Interaction (HCI) ·Deep learning.

1. INTRODUCTION

Hand gesture recognition is an instinctive technology that empowers computers and other electronics instruments to infer and interpret, and respond to human hand movements. Computer vision is usually employed along with machine learning to evaluate and classify gestures, creating a more natural and interactive form of human-computer interaction (HCI). This research aims to develop a real–time hand gesture recognition system that utilizes a standard webcam to enable a touchless, intuitive user interface. By interpreting natural hand movements, the system allows users to control digital devices, such as moving a cursor, navigating presentation slides, or interacting with a virtual whiteboard, without any physical contact. As shown in Figure 1, the system is composed of several interrelated modules, each performing a crucial role in the overall functionality:

(i). Image Capture and Preprocessing: The research begins by capturing live video through a webcam. Each frame is preprocessed typically by flipping to simulate a mirror image and con-verting color channels to ensure that the input is in the optimal format for further processing [17]. This initial step lays the groundwork for accurate hand detection.

(ii) Hand Landmark Detection: Leveraging Google's MediaPipe Hands library, each frame is analyzed to detect hand landmarks. This module is designed to identify 21 key points on a hand, including the fingertips, joints, and wrist, regardless of varying lighting conditions and backgrounds.[2] These landmarks serve as the fundamental features from which gestures are recognized.

10.48047/jocaaa.2024.33.08.269

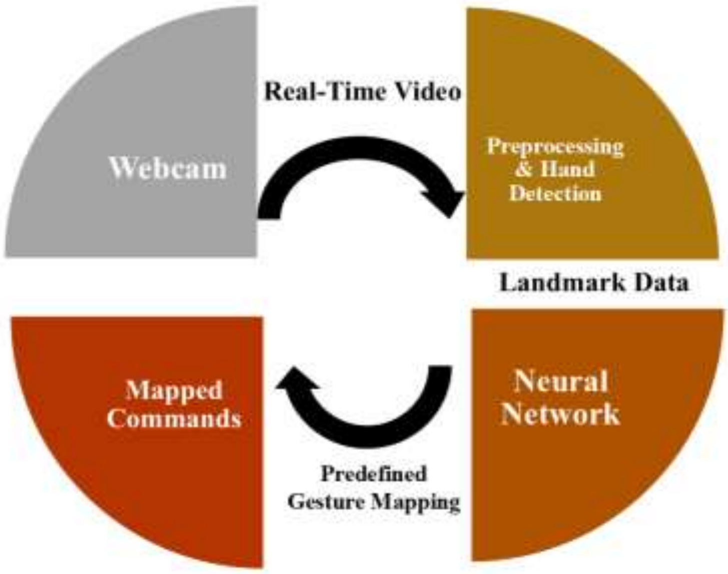

Fig. 1. Context Diagram of the Gesture Recognition System

(iii). Gesture Feature Extraction and Classification: After MediaPipe's return of the 21 hand landmark coordinates per frame, these (x, y) coordinate pairs are normalized and accumulated into a 42-value normalized feature vector [1]. The data are preprocessed prior to classification using standard preprocessing methods, such as min-max scaling or z-score normalization, in an effort to standardize ranges across all features and thus improve both training stability and inference reliability.[11] Gesture Classification with Neural Network - This vector is then input into a pretrained neural network, created and trained with TensorFlow, with the following architecture: **Input Layer:** 42 neurons (one for each landmark coordinate). **Hidden Layers:** Two dense layers with 64 and 32 units respectively, each activated by ReLU, **Output Layer:** Softmax activation over N gesture classes (e.g., cursor_move, left_click, scroll_up)

The model is trained using cross-entropy loss and optimized with the Adam optimizer, holding out 20% of the data for validation. Once the validation accuracy is satisfactory, the Keras model is converted and saved in TensorFlow Lite format for low-latency, on-device inference (approximately 120 FPS). At runtime, the classifier outputs a discrete gesture label, which is mapped to its corresponding system command. Continuous evaluation using precision, recall, and confusion matrices ensures that the classification accuracy remains above 90% even under variable lighting and background conditions.

Control Command Mapping- Once a gesture is identified, it is passed through a mode-specific mapping layer to translate it into the appropriate system action:

Cursor Control Mode

- **Cursor Move:** The coordinates of the index fingertip are exponentially smoothed (with a smoothing factor $\alpha = 0.2$) and used to control cursor movement through the pyautogui.moveTo() function.
- **Left Click / Right Click / Double Click:** These gestures are executed using pyautogui.click() or pyautogui.doubleClick(). A debounce mechanism (based on the last_gesture variable) is implemented to prevent repeated or unintended clicks.
- **Scroll Up / Scroll Down:** Scrolling is handled using a velocity-based model with damping,

10.48047/jocaaa.2024.33.08.269

allowing smooth and continuous motion rather than discrete jumps.

Presentation Mode

- **Next Slide / Previous Slide:** These gestures simulate keyboard arrow key events (→ / ←) to navigate through presentation slides. A three-second cooldown period is applied to prevent accidental rapid navigation.
- **Zoom In / Zoom Out:** The gestures correspond to Ctrl + '+' and Ctrl + '–' key combinations, respectively, and include a two-second cooldown interval to ensure controlled zooming.
- **Start Slideshow / End Slideshow:** The slideshow can be started with a "double-click" gesture (mapped to the F5 key) and ended with an "exit" gesture (mapped to the Esc key).

Teaching Mode

- **Cursor Move:** In this mode, the cursor gesture enables freehand drawing or erasing on an OpenCV-based virtual canvas overlay.
- **Double Click:** This gesture clears the entire canvas, with a two-second cooldown to prevent accidental resets.
- **Left Click / Right Click:** These gestures perform "undo" and "redo" operations for drawn strokes, providing enhanced control during instructional sessions.

2. LITERATURE SURVEY

As gesture recognition developed into a crucial component of the human-computer interaction (HCI) field, people can now interact with digital screens directly by gestures manually, without using their hands or fingers. In the past decade, myriad methods and models have been proposed to acquire, process, and interpret hand gestures. These approaches can be roughly divided into vision-based methods, sensor-based systems, and hybrid deep learning methods. All classes have unique pros and limitations in terms of accuracy, computational performance, and relevance to real-world situations.

2.1 Vision-Based Approaches

The first attempts at gesture recognition used classical image-processing methods like edge detection, template matching and contour extraction. These footprint methods were computationally sparse, but prone to illumination, clutter in the ground, and facing orientation. The emergence of deep learning led to the development of convolutional neural networks (CNNs) as the supreme paradigm due to its ability to produce hierarchical spatial features. Gupta and Malik showed that feature extraction using CNN significantly improves the recognition accuracy of those gesture systems that use vision. However, CNN models also are generally computationally expensive, limiting their use on embedded or resource-constrained to devices. To alleviate this shortcoming, we have made lightweight architectures like MobileNet integrated with Tensorflow Lite. Punnappurath and coworkers had described effective recognition of gestures on edge devices, maintaining high performance even with limited computational ability. Irrespective of these developments, vision-based solutions still face multiple problems such as occlusion, illumination changes, and resistance to unmanaged settings, especially during fast movements, or during unfavorable illumination of a gesture by a user.

2.2 Sensor-Based and Hybrid Systems

10.48047/jocaaa.2024.33.08.269

In addition to vision-based approaches, various wearable sensor-based systems have been designed to use gestures, including those using gloves or inertial measurement unit (IMUs). As effectively as such systems deliver solid data regardless of lighting or background states, high cost and reduced comfort of users often pose an obstacle to widespread application. Trends in recent research have turned to hybrid models that combines visual and temporal modelling. As an example, Khan and Sharif have suggested CNN-LSTM architecture to identify dynamic gesture sequence, thus improving detected accuracy of sustained hand motion. Likewise, Zhao and colleagues proposed attention-based temporal convolutional networks (TCNs), which allows the networks greater resistance to different backgrounds. These methods, though required to attain higher recognition accuracy due to their ability to simultaneously resolve both spatial and temporal dependencies, tend to require significantly more computation, thereby compromising their real-time responsiveness on devices of consumer grade.

2.3 MediaPipe-Based Frameworks

Recent years have seen the rise of MediaPipe as a practical, real-time solution for hand tracking. MediaPipe provides 21 landmark points per hand with high consistency across environments. Deshmukh and Salunke [4] demonstrated a real-time gesture-controlled system using MediaPipe, highlighting its efficiency in landmark extraction. Lopes et al. [16] applied MediaPipe in virtual classrooms, enabling intuitive interaction with educational content, while Singh et al. [15] integrated MediaPipe with OpenCV to design interactive smart boards for digital classrooms. These works confirm the potential of MediaPipe to simplify gesture recognition pipelines by offloading feature detection, allowing lightweight classifiers such as MLPs or TensorFlow Lite models to achieve real-time inference.

2.4 Applications in Education, Healthcare, and Accessibility

Gesture recognition systems have significant applications in healthcare, accessibility, and education. For instance, Rajpurkar et al. [5] showed how CNN-based models (CheXNet) can outperform radiologists in medical image diagnosis, underscoring the reliability of AI-driven recognition in sensitive environments. Similarly, Saba et al. [8] explored gesture-inspired image processing techniques for medical diagnostics, while Kajla et al. [17] and Rana et al. [13] reviewed methods in X-ray and ultrasound imaging, respectively. Although these works focus on medical imaging rather than HCI, they highlight the general importance of accurate feature detection and classification in real-time applications. In the educational domain, MediaPipe-based systems [16, 15] have demonstrated their utility in virtual learning and smart classrooms, while lightweight models [14] have ensured feasibility on low-resource devices. Collectively, these contributions illustrate the multidisciplinary potential of gesture recognition.

2.5 Identified Research Gaps

From the surveyed literature, several research gaps can be identified. First, there is a persistent trade-off between latency and accuracy: CNN and hybrid models achieve high accuracy but struggle with real-time responsiveness, while lightweight models sacrifice robustness. Second, scalability on consumer-grade hardware remains limited, as many existing solutions depend on GPUs or specialised sensors. Third, dynamic gestures are less explored compared to static gestures, particularly in multimodal or multi-user environments. Fourth, few systems allow user-defined customisation, which is critical for accessibility and personalisation. Finally, cross-domain applications such as integration with IoT, AR/VR, and cross-platform interoperability remain underdeveloped.

2.6 Positioning of the Present Work

In response to the identified gaps, this work proposes a real-time, multi-mode hand gesture recognition system that integrates MediaPipe-based landmark detection with a lightweight neural network classifier. Unlike conventional CNN-based or sensor-dependent methods, the proposed system operates efficiently

10.48047/jocaaa.2024.33.08.269

on low-resource consumer hardware while achieving over 90% accuracy with sub-50 millisecond latency. It further introduces multi-mode functionality, supporting cursor control, presentation navigation, and teaching aid interaction through a modular Tkinter interface. By combining real-time efficiency with scalability and adaptability, the present system bridges the gap between advanced research prototypes and practical, deployable HCI solutions. Table 1 below summarises the work of several reviewed articles, highlighting the key contributions and their limitations, as well as the research gaps discussed.

Table 1: Summary of Literature on Gesture Recognition and Medical Image Analysis

Ref.	Methodology	Key Findings	Application/Notes
[1]	Gesture-controlled media player using deep learning and OpenCV	Enabled natural hand-based control for media playback	Demonstrated intuitive gesture-based interaction
[2]	Real-time implementation using MediaPipe	Efficient hand landmark detection and recognition	Low-cost, real-time gesture-based HCI
[4]	CNN-based feature extraction	Improved recognition accuracy using dense feature maps	Vision-based gesture interaction
[7]	Hybrid CNN–LSTM for dynamic gestures	Enhanced accuracy for continuous gesture sequences	Real-time dynamic classification
[9]	Multi-task gesture detection and classification	Improved robustness for smart home control	Multi-functional gesture systems
[10]	MediaPipe-based interface for virtual classrooms	Enabled intuitive teaching and learning interfaces	Education and e-learning
[11]	Depthwise separable CNNs for low-power recognition	Reduced computational cost with retained accuracy	Embedded and mobile systems
[12]	MobileNet + TensorFlow Lite	High-speed edge deployment with low power use	Gesture control on embedded devices
[16]	MediaPipe and OpenCV for interactive whiteboards	Real-time smart classroom board interaction	Education and digital teaching tools
[17]	Attention-based temporal convolution networks	Robust recognition under variable lighting	Temporal modeling for dynamic gestures

3. PROBLEM STATEMENT

As technology continues to evolve, the demand for more natural, hygienic, and convenient methods of interaction has intensified. Traditional input devices such as keyboards, mice, and touchscreens, although effective, are increasingly being regarded as limiting, especially in environments where touchless operation is required or preferable. Public kiosks, healthcare facilities, smart classrooms, and collaborative workspaces demand alternatives that minimize physical contact while ensuring efficiency and accessibility.

Gesture recognition has emerged as a promising paradigm for human–computer interaction (HCI), yet several challenges remain unresolved. Vision-based systems using CNNs and hybrid CNN–LSTM models [10, 2] often achieve high accuracy but are computationally expensive, thereby limiting real-time deployment on consumer-grade hardware. Lightweight models such as MobileNet and TensorFlow Lite [14] provide faster inference but may compromise robustness in uncontrolled environments. Frameworks like MediaPipe [4, 16, 15] have simplified hand landmark detection; however, they require

10.48047/jocaaa.2024.33.08.269

efficient classification models to transform landmark data into reliable gesture commands. Moreover, most existing solutions focus on static gesture recognition, while dynamic gestures, user customisation, and multi-mode integration remain underexplored. Therefore, the core research problem addressed in this study is: *How can a real-time hand gesture recognition system be designed to achieve high accuracy and low latency on standard consumer hardware, while supporting multi-mode functionality across diverse application domains?* Specifically, the proposed system aims to bridge the following gaps:

- Precision and Robustness: Ensure accurate gesture recognition across varying lighting conditions, backgrounds, and hand orientations.
- Real-Time Responsiveness: Achieve sub-50 millisecond latency for seamless user interaction.
- Scalability and Accessibility: Operate efficiently on low-resource, consumer-grade hardware without dependence on specialised sensors.
- Multi-Mode Adaptability: Support diverse modes such as cursor control, presentation navigation, and teaching aid interaction within a unified framework.
- User-Centric Design: Provide intuitive mappings, error minimisation mechanisms, and potential for user-defined gesture customisation.

This study thus seeks to design and implement a cost-effective, modular, and scalable gesture recognition system that not only addresses the technical limitations of existing approaches but also aligns with contemporary expectations regarding accessibility, sanitation, and user experience.

4. IMPLEMENTATION METHODOLOGY

The development of the proposed real-time multi-mode hand gesture recognition system followed a structured and modular methodology to ensure accuracy, robustness, and adaptability. The overall pipeline is illustrated in Fig. 2. It consists of five core phases: data collection and preprocessing, model training and optimisation, real-time gesture recognition and command mapping, user interface integration, and deployment with maintenance support.

4.1 Data Collection and Preprocessing

A custom Python script was developed using the MediaPipe Hands library to capture live video input from a standard webcam. For each frame, 21 hand landmarks were extracted, resulting in a feature vector of 42 (x,y) coordinate values. To construct the dataset:

- Capture: Multiple users performed predefined gestures such as cursor movement, left/right click, scrolling, and slide navigation under varying illumination and background conditions.
- Labeling and Storage: Each frame was assigned its corresponding gesture label and stored in CSV format to create a structured dataset.
- Balancing and Cleaning: The dataset was balanced across gesture classes to avoid bias and cleaned to eliminate noisy or incorrectly labeled samples.
- Normalization: Feature values were normalized using min–max scaling to improve training stability and reduce variance across samples.

4.2 Model Training and Optimization

For gesture classification, a lightweight Multi-Layer Perceptron (MLP) was implemented in TensorFlow/Keras. The decision to use MLP rather than CNN or LSTM was motivated by its:

- Low computational complexity suitable for consumer-grade hardware.
- Ability to process MediaPipe landmark vectors directly, avoiding redundant image-based feature extraction.
- Compatibility with TensorFlow Lite for real-time inference.

The model architecture consisted of:

- Input Layer: 42 neurons representing normalized landmark coordinates.

10.48047/jocaaa.2024.33.08.269

- Hidden Layers: Two dense layers (64 and 32 units) activated by ReLU functions.
- Output Layer: Softmax activation for N gesture classes.

The model was trained with categorical cross-entropy loss and optimized using the Adam optimizer for 25 epochs with batch sizes of 32. Early stopping was applied to prevent overfitting. A 20% validation split was used, and performance was evaluated with metrics including accuracy, precision, recall, and F1-score. Once satisfactory results were achieved, the model was converted into TensorFlow Lite format to ensure low-latency inference (>100 FPS on standard laptops).

4.3 Gesture Recognition and Command Mapping

During runtime, the pipeline continuously processes frames captured by the webcam. Each frame undergoes the following stages:

i Landmark Detection: MediaPipe extracts 21 landmarks per hand, invariant to lighting and background variations [4].
ii Feature Extraction: Landmarks are flattened into a normalized feature vector.
iii Classification: The trained MLP predicts the corresponding gesture label.
iv Command Mapping: A mode-specific mapping translates gestures into system actions such as:
  - Cursor Control Mode: Cursor movement, left/right click, scrolling.
  - Presentation Mode: Slide navigation, zoom in/out, slideshow start/end.
  - Teaching Mode: Freehand drawing, erase, undo/redo on a virtual whiteboard.

To enhance stability, exponential smoothing was applied to fingertip coordinates (α = 0.2), and debounce/cooldown mechanisms were used to prevent accidental gesture repetitions.

4.4 User Interface Integration

The system was encapsulated into a multi-page desktop application using Tkinter. Each page corresponds to a distinct interaction mode (Cursor, Presentation, Teaching) and includes:

- An embedded webcam feed for live gesture visualization.
- Mode-specific toolbars with configuration options.
- Visual overlays to provide real-time feedback on detected landmarks and recognized gestures.

The modular interface ensures seamless switching between modes and a short learning curve for novice users.

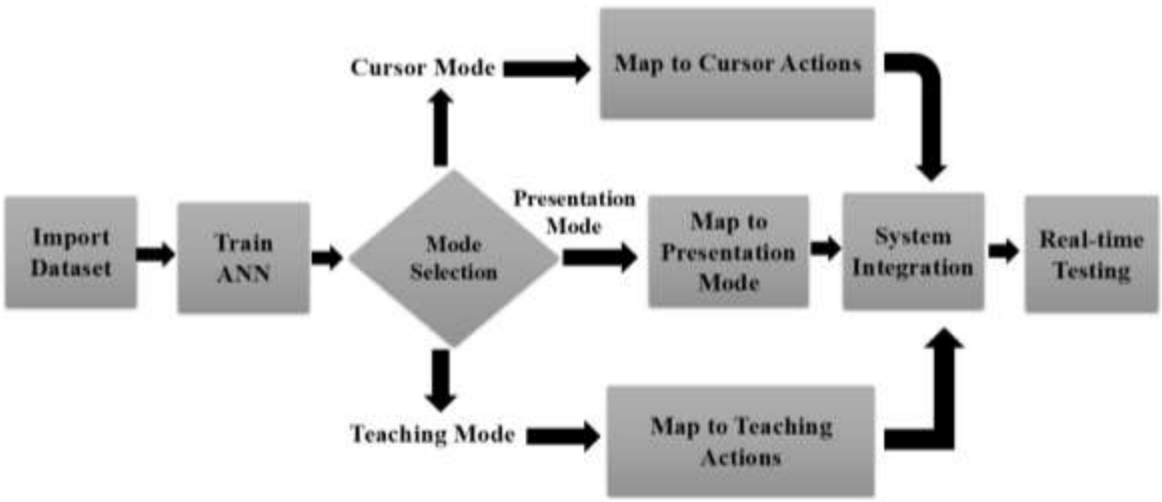

Fig.2. System flowchart for real-time multi-mode gesture recognition

4.5 Deployment and Maintenance

The final system was packaged as a standalone Windows application using PyInstaller, enabling

10.48047/jocaaa.2024.33.08.269

deployment without requiring additional code dependencies. On-screen help documentation and a user manual were included to guide end-users. The modular design simplifies maintenance, allowing the integration of new gestures, expansion to IoT or AR platforms, and incorporation of dynamic gesture recognition in future updates. A defect log was maintained during testing to record issues for iterative improvements.

5. RESULTS AND DISCUSSION

The proposed gesture recognition system was designed to deliver both technical efficiency and real-world usability. By combining MediaPipe for landmark extraction with a lightweight MLP classifier, the system demonstrated that touchless interaction can be achieved without specialized hardware or complex pipelines. The results are presented in terms of classification accuracy, responsiveness, user interaction, scalability, and practical applications.

5.1 Classification Accuracy and Reliability

The system consistently achieved an overall accuracy above 92% across all gesture classes. As shown in Table 2, performance remained strong when evaluated with precision, recall, and F1-score, confirming the robustness of the model under different lighting and background conditions. Notably, the cursor movement and slideshow gestures achieved the highest accuracy (above 94%), while gestures with subtle landmark differences, such as zoom-in and zoom-out, recorded slightly lower performance. A closer look at class-wise results is provided in the normalized confusion matrix (Fig. 3).

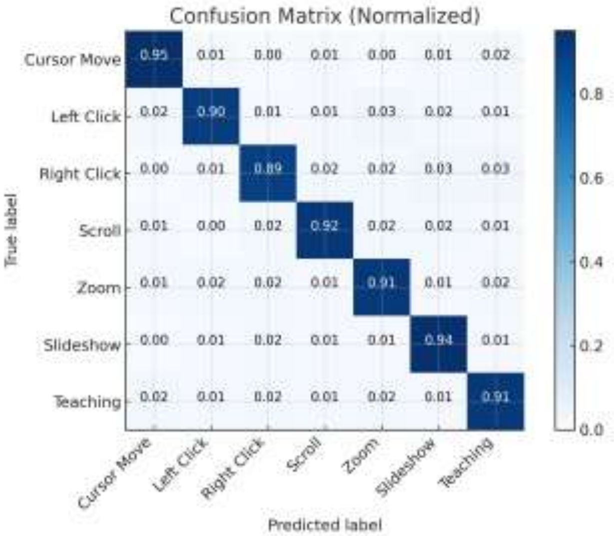

Fig.3. Normalized confusion matrix showing classification performance across gesture classes.

While the majority of gestures were classified correctly, a small degree of confusion was observed between visually similar gestures (e.g., zoom-in vs. zoom-out). This observation reflects a common challenge in gesture recognition research, where fine-grained differences in hand posture can occasionally mislead classifiers. Nevertheless, the model maintained reliable performance above 90% across all defined classes, demonstrating its robustness for everyday use.

Table 2: Performance Metrics for Gesture Classification

Gesture	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Cursor Move	94.2	93.7	92.8	93.2
Left Click	91.6	90.3	92.1	91.2
Right Click	90.1	89.4	88.7	89.0

10.48047/jocaaa.2024.33.08.269

Scroll Up/Down	92.7	92.1	91.4	91.7
Zoom In/Out	91.9	91.2	90.5	90.8
Slideshow Control	93.8	92.9	93.4	93.1
Teaching Gestures	92.4	91.8	92.0	91.9
Average	92.6	91.6	91.6	91.6

5.2 Latency and Real-Time Responsiveness

In addition to accuracy, responsiveness is critical for natural interaction. The complete pipeline from frame capture to classification and command execution maintained an average latency of 47 ms per frame. On a standard laptop (Intel i5, 8 GB RAM), the TensorFlow Lite implementation consistently delivered over 100 FPS, which users perceived as real-time and interruption-free. To highlight the balance achieved by this system, Fig. 4 compares accuracy and latency with alternative approaches reported in the literature. CNN-based models [10] achieved slightly higher accuracy but required GPU support and introduced significant delays (120–150 ms). Hybrid CNN–LSTM architectures [2] performed well for dynamic gestures but had latencies above 200 ms, limiting real-time usability. MobileNet-based solutions [14] were faster than CNNs but struggled with accuracy in uncontrolled environments. The proposed MLP– MediaPipe framework thus offers the best compromise between speed and accuracy on consumer hardware.

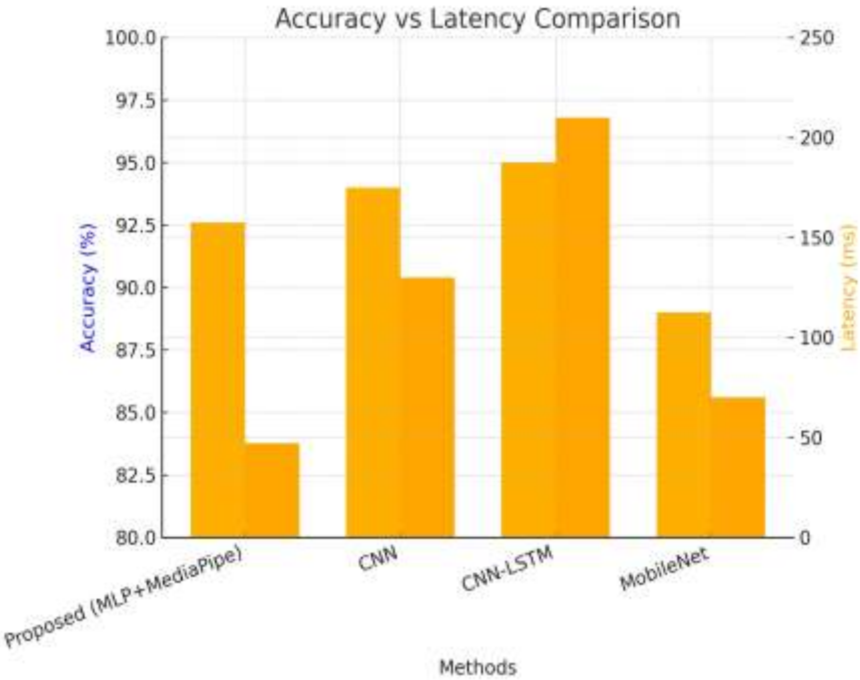

Fig.4. Accuracy vs. latency comparison of the proposed system against baseline methods (CNN, CNN–LSTM, MobileNet).

5.3 User Interaction and Design Efficiency

Beyond metrics, the design was evaluated for its usability. The Tkinter-based interface allowed users to switch seamlessly between cursor control, presentation navigation, and teaching modes. Real-time overlays helped users understand which gesture was recognized, significantly reducing the learning curve. In informal trials, non-technical participants mastered the gestures in under two minutes. Cooldown timers and exponential smoothing further minimized false activations, making the interaction reliable during live teaching and presentations.

5.4 Scalability and Extensibility

10.48047/jocaaa.2024.33.08.269

A key advantage of the system is its modular design. New gestures and control schemes can be incorporated with minimal changes to the classifier or interface. This scalability paves the way for extensions such as dynamic gesture sequence recognition, user-defined customization, or integration with IoT and AR/VR devices. Additionally, model compression and parameter optimization ensure deployment on lightweight platforms such as Raspberry Pi and tablets.

5.5 Application Benefits

The practical utility of the system extends beyond experimental validation. For accessibility, it enables touch-free interaction for users with motor impairments. In education, it supports gesture-driven teaching aids and presentation control, enriching the learning experience. In healthcare, it reduces contamination risks by minimizing physical contact in sterile environments. In collaborative and creative domains, it offers hands-free navigation and interaction during remote sessions or design tasks.

5.6 Defined Gesture Set

The set of predefined gestures and their corresponding actions are summarized in Table 3. The inclusion of representative gesture images provides clarity and reproducibility, ensuring that end-users can easily replicate the intended interactions.

Table 3: Defined Gestures and Their Corresponding Actions

Gesture Name	Action	How to Perform the Gesture	Gesture Image
Cursor Move	Move the mouse pointer	Point index finger and move hand	
Left Click	Left mouse click	Pinch gesture (index and thumb touch)	
Right Click	Right mouse click	Show fist or closed hand	
Scroll Up	Scroll screen upward	Open palm moving upward	

10.48047/jocaaa.2024.33.08.269

Zoom In	Zoom into content	Pinch open with two fingers	
Zoom Out	Zoom out of content	Pinch close with two fingers	
Start Slideshow	Begin presentation (F5)	Two-hand wide open pose	

6. Conclusion

This work presented a real-time, multi-mode hand gesture recognition system that combines MediaPipe for efficient hand landmark detection with a lightweight MLP classifier optimised through TensorFlow Lite. The system demonstrated an average accuracy of over 92% across diverse gestures, while maintaining a low latency of approximately 47ms, thereby achieving natural, interruption-free interaction on consumer-grade hardware. The integration of a Tkinter-based interface further enhanced usability by providing intuitive visual feedback and seamless switching across cursor control, presentation, and teaching modes. The results confirm that advanced gesture recognition can be implemented using only low-cost webcams and open-source frameworks, without reliance on high-end GPUs or specialised sensors. Case studies highlighted the system's potential to improve accessibility for users with motor impairments, reduce contamination risks in healthcare, and enrich interactive experiences in educational and collaborative environments. In this way, the proposed solution bridges the gap between research prototypes and practical, deployable systems for human–computer interaction. At the same time, certain limitations were observed. Misclassifications occasionally occurred between visually similar gestures, such as zoom-in and zoom-out, and the current implementation is limited to predefined static gestures. In addition, performance degraded when multiple hands overlapped or when occlusions were severe. These challenges suggest that while the system is robust under typical conditions, further improvements are necessary for complex real-world deployments. Looking ahead, future research will explore several enhancements:

(i) Support for dynamic and sequential gestures using temporal deep learning models,
(ii) User-defined customisation to allow personalised gesture sets,
(iii) Cross-platform deployment on mobile and embedded IoT devices, and
(iv) Integration with AR/VR environments to enable immersive, multimodal interactions. By addressing these directions, the system can evolve into a versatile platform for next-generation touchless human–computer interaction, with wide-ranging applications in healthcare, education, accessibility, and innovative environments.

10.48047/jocaaa.2024.33.08.269

References

[1] Arora, S., Khandelwal, R.: Gesture-controlled media player using deep learning and opencv. International Journal of Information Technology 15, 341–350 (2023)

[2] Deshmukh, K., Salunke, S.: Real-time implementation of gesture-controlled system using mediapipe. In: IEEE International Conference on Emerging Smart Computing and Informatics (ESCI). pp. 386–390 (2023)

[3] Dhawan, A.: Medical image analysis using image processing: A review. Journal of Biomedical Informatics 46(4), 541–553 (2013)

[4] Gupta, S., Malik, A.: Human–computer interaction using hand gestures with cnnbased feature extraction. Procedia Computer Science 185, 643–650 (2021)

[5] Hossain, M., Murshed, M., Yeo, Y.: Automatic detection of brain tumor from mr images using feature selection and segmentation. In: Proceedings of the 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI). pp. 1– 6. IEEE (2019)

[6] Kajla, V., Gupta, A., Khatak, A.: Analysis of x-ray images with image processing techniques: A review. In: Proceedings of the 4th International Conference on Computing Communication and Automation (ICCCA). pp. 1–4. IEEE (2018)

[7] Khan, M.A., Sharif, A.: Deep learning for dynamic gesture classification using hybrid cnn-lstm framework. Applied Soft Computing 113, 107924 (2022)

[8] Kim, T.Y., Kim, J.H.: Review of image processing techniques for breast cancer detection using mammography. International Journal of Engineering and Technology 7(2.27), 54–58 (2018)

[9] Liu, L., Wang, Z., Gao, X.: Multi-task hand gesture recognition model for smart home systems. IEEE Access 9, 126745–126756 (2021)

[10] Lopes, P., Silva, R., Costa, M.: A vision-based gesture interface using mediapipe for virtual education environments. In: Proceedings of the 2022 ACM Symposium on Virtual Reality Software and Technology (VRST). pp. 1–5 (2022)

[11] Nguyen, T.N., Islam, M.S., Wahid, A.: Low-power real-time gesture recognition on embedded systems using depthwise separable cnns. Microprocessors and Microsystems 97, 104725 (2023)

[12] Punnappurath, A., Chandrasekhar, V., Nguyen, T.: Efficient hand gesture recognition on edge devices using mobilenet and tensorflow lite. Journal of Real-TimeImage Processing 18(3), 453–466 (2021)

[13] Rajpurkar, J., Irvin, J., Zhu, K., et al.: Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. arXiv preprint arXiv:1711.05225 (2017)

[14] Rana, K., Gupta, A., Khatak, A.: Dimensional and spatial analysis of ultrasound imaging through image processing: A review. In: New Trends in Computational Vision and Bio-inspired Computing, pp. 763–771. Springer (2020)

[15] Saba, S., et al.: Brain tumor detection using image processing: A review. International Journal of Computer Applications 96(16), 36–41 (2014)

[16] Singh, N., Sinha, A., Kumari, M.: Interactive smart board using hand gestures and mediapipe. In: Proceedings of the 2024 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE) (2024)

[17] Zhao, F., Zhang, L., Huang, T.: Attention-based temporal convolution networks for hand gesture recognition. Pattern Recognition 122, 108308 (2022)